Efficient Multi-Agent Exploration with Gaussian Processes

Alberto Viseras Ruiz*, **Michael Angermann,**
Iris Wieser, Martin Frassl, Joachim Mueller
Institute of Communications and Navigation
German Aerospace Center (DLR)
Wessling, Germany
*alberto.viserasruiz@dlr.de

Abstract

We present a robust and scalable algorithm to enable multiple robots to efficiently explore previously unknown environments. Applications of this algorithm include but are not limited to the exploration of scalar (e.g. concentration of a chemical substance) or vector fields (e.g. direction and intensity of the magnetic field). As opposed to previous works, our algorithm does not require prior knowledge about the shape or size of the environment. Also, its time complexity decreases from cubic to linear in respect to the environment's size. The algorithm employs a Gaussian process model to predict values at still unvisited locations and associates them an uncertainty. Based on a continuously updated map of these predicted values and uncertainties, each robot computes its next movement online by following the local gradient of uncertainty. We have experimentally validated our algorithm with densely measured data of an indoor magnetic field with significant spatial variations. We present a performance comparison of our algorithm with several trajectories. This comparison shows that the estimate computed by our algorithm approaches the true state of the environment faster than these other alternatives.

1 Introduction

Exploration is a fundamental task in a wide range of applications, such as prospecting, environmental analysis, or search and rescue. Often mobile robots are well-suited to carry sensors to locations at which sampling should take place. In general it is desirable to reduce the required time and manpower for the exploration of a given environment. This naturally leads to parallelization by means of multiple robots, which form a self-coordinating multi-agent system with relatively little human supervision per robot.

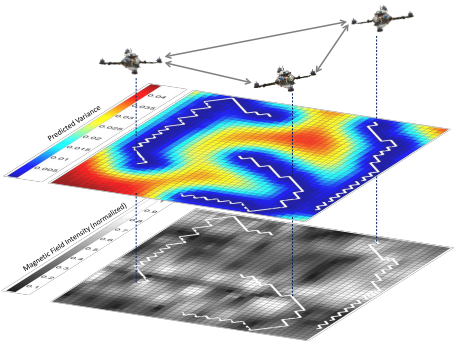


Figure 1: Multi-agent system exploring a vector field (here magnetic field intensity) with the Sense-Predict-Move algorithm. The agents sense the physical process and broadcast the measurements. They predict the remaining values employing a Gaussian process model. The agents calculate their new position optimizing a reward function, which makes them follow gradients of uncertainty. On top, we show the variance predicted by the Gaussian process as the agents move. The white lines represent the agents' trajectories.

1.1 Motivation

While brute-force systematic sampling may often be feasible, we are motivated to derive algorithms that, without previous knowledge of the environment, allow us to obtain desired exploration results more rapidly, and/or with less resources. This may be economically advantageous or even life-critical in search and rescue missions. In particular, we are interested in obtaining a usable map with sufficient information in the least amount of time. We aim to derive a low computational complexity algorithm, such that it could be applied to a mission with small robots like Quadrocopters. To obtain the most informative samples in a minimum amount of time, we want to exploit spatial dependencies of the region of interest and model its parameters. This also allows us to predict desired information of places that have not been visited yet to obtain faster a usable map of exploration

results.

1.2 Problem Statement

We wish to explore an environment as accurately as possible, in the sense of minimizing the difference between estimate and ground truth, and as efficiently, in the sense of consuming as little as possible of the limited resources, such as time, energy, or communication capacity.

Without loss of generality we formulate the problem, model and algorithm for a two-dimensional case in order to avoid notation clutter. A scenario is composed of an *environment*, *information objects* and N_A *agents*.

The *environment* S is the physical space in which the information objects and the agents are located. Let $s_{r,c} \in \mathbb{R}^2$ be the central position of the cell (r, c) in an $R_g \times C_g$ grid. We define the environment as $S = [s_{r,c}]_{r=1,...,R_g;\ c=1,...,C_g}$.

Each information object $o_{r,c} \in \mathbb{R}$ is a Gaussian distributed random variable located at the spatial position $s_{r,c}$. The set of all *information objects* is defined as $O = [o_{r,c}]_{r=1,...,R_g;\ c=1,...,C_g}$.

The *agent's* network is represented by a set of identifiers $I = \{1, ..., N_A\}$ and a set of robots $R = \left\{R^{[i]}\right\}_{i \in I}$.

A robot $R^{[i]}$ stores: its spatial location $p^{[i]} \in S$, its set of measurements $Z^{[i]}$, the combined map of predictions and measurements $P^{[i]} = \left[P^{[i]}_{r,c}\right]_{r=1,...,R_g;\ c=1,...,C_g}$ and the variance map $U^{[i]} = \left[U^{[i]}_{r,c}\right]_{r=1,...,R_g;\ c=1,...,C_g}$.

The presence of obstacles in the environment is not considered in this work. We also assume that there is a permanent and perfect communication link between any two agents in the environment.

1.3 Related Work

There is a growing interest in multi-agent systems which cooperate to achieve a common objective [Olfati-Saber *et al.*, 2007]. Our focus is mapping physical phenomena with a team of robots. To monitor environments, mobile sensors have been proven to be very efficient [Howard *et al.*, 2002; Ogren *et al.*, 2004; Poduri and Sukhatme, 2004]. Our goal is not only monitoring, but obtaining a fine usable representation of a static physical phenomena within a previously unknown environment. Furthermore, to increase the algorithm's efficiency, we aim to measure only a subset of points in the environment and predict the remaining values. Therefore, we exploit the spatial dependencies of the process under study [Rachlin *et al.*, 2005].

Markov random fields have been used to derive a level curve tracking algorithm to map an oceanographic process with a multi-agent system [Williams and Sukhatme, 2012]. For the experimental setup, the variables under study are modeled as binary processes to reduce complexity. However, we are interested in continuous valued variables. Also, in order to take advantage of the process' spatial structure, we consider more complex probabilistic models.

Gaussian Processes are one of the most common methods to model physical phenomena [Rasmussen and Williams, 2005]. Fink and Kumar propose in [Fink and Kumar, 2010] an online estimation of a radio signal source based on a Gaussian Process model of the propagated signal. However, they only consider the single-robot case. Also, the computational complexity grows cubically with the number of predictive samples and therefore, does not allow big environments.

In [Stranders *et al.*, 2008], Stranders *et al.* propose a decentralized algorithm for online monitoring of spatial phenomena using several mobile sensors. They model an isotropic physical process as a Gaussian Process with a Matérn covariance function. The sensors' motion follows an entropy maximization criterion. However, as the sensors placement is restricted, the environment can not be explored with complete freedom and is limited in size by the amount of agents. In contrast to the last two approaches, we propose a multi-agent exploration algorithm that is scalable with the environment's size.

Krause *et al.* present in [Krause and Guestrin, 2007] a single-agent exploration algorithm using a Gaussian Process model. The core contribution of their work is the proof of a theoretical bound which quantifies the differences between active (the agent calculates its new position based on the observations) and a priori strategies. This algorithm is extended to the multiple robots case by Singh *et al.* in [Singh *et al.*, 2009]. They propose an offline algorithm to compute quasi-optimal paths of a team of robots to maximize the collected information given their start and finishing position. They use an underlying Gaussian Process model and consider that the robots have bounded resources. A cell spatial decomposition of the environment handles the algorithm's complexity. But, for a cell spatial decomposition as well as a definition of the starting and final location, previous information of the environment is needed. Moreover, the algorithm is not robust against failure of agents. In this work, we derive a distributed online algorithm that allows robot failure. Also, it does not require a prior knowledge about the shape or size of the environment.

Motivated by research on using the magnetic field inside buildings for indoor localization [Angermann *et al.*, 2012], we chose to validate our algorithm with the exploration of the indoor magnetic field as our physical process under study.

The remainder of the paper is organized as follows. In Section 2 we introduce the theoretical model on which the algorithm is based and give an overview of the Gaussian Processes for regression. In Section 3 we introduce

the Sense-Predict-Move algorithm in detail, presenting two different reward functions to control the robots' motion. In Section 4 we describe our experimental setup and results. We further present the results of simulations to investigate scalability and robustness and compare the performance of our algorithm to several trajectories. We discuss our findings in Section 5.

2 Spatial Gaussian Process Model

Consider, for example, the task of obtaining a map of the magnetic field intensity in an indoor environment. If we could model the spatial dependencies of the magnetic field well, we could fill spatial gaps between measurements with predictions [Cressie, 1992]. The stronger the dependencies and the better they are represented in a model, the less measurements we need to achieve a certain accuracy. Gaussian Processes represent an alternative to model physical phenomena with strong spatial variations; e.g. temperature, magnetic field intensity [Kemppainen *et al.*, 2010], etc.

A Gaussian Process [Rasmussen and Williams, 2005] is a collection of random variables, any finite number of which have a multivariate Gaussian distribution. It is fully specified by a mean function $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$.

We define the following vectors[1]: 1) $\mathbf{x}= [x_1, x_2, ..., x_n]$ is the n-observations input space. 2) $\mathbf{f}= [f_1, f_2, ..., f_n]$ is the set of the observed target values. Each of the observed target values corresponds to one of the variables in the n-observations input space. 3) $\mathbf{x}_*$ is the predictive input space.

Gaussian Processes are commonly used as priors in a Bayesian setting. Given $\mathbf{f}$ and $\mathbf{x}$, we can predict the target values $\mathbf{f}_*$ for the predictive input space $\mathbf{x}_*$. The elements in $\mathbf{f}_*$ are distributed according to: $p(\mathbf{f}_*|\mathbf{x}_*, \mathbf{x}, \mathbf{f}) = \mathcal{N}(\bar{\mathbf{f}}_*, \sigma^{\mathbf{2}}_{\mathbf{f}_*})$. The mean vector $\bar{\mathbf{f}}_*$ and the variance vector $\sigma^{\mathbf{2}}_{\mathbf{f}_*}$ of the posterior distribution are calculated as:

$$\begin{aligned} \bar{\mathbf{f}}_* &= m(\mathbf{x}_*) + K_*^T \cdot K^{-1} \cdot (\mathbf{f} - m(\mathbf{x})), \\ \sigma^{\mathbf{2}}_{\mathbf{f}_*} &= K_{**} - K_*^T \cdot K^{-1} \cdot K_*. \end{aligned} \tag{1}$$

The matrices K, K_*, K_{**} are defined from the covariance function as:

$K = [k(x_i, x_j)]_{i,j=1,...,n}$, $K_* = [k(\mathbf{x}_*, \mathbf{x_i})]_{i=1,...,n}$, $K_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$.

The definition of the covariance function assumes the notion of similarity, which means that we expect that closer points are more likely to be similar. We focus our interest in stationary and isotropic covariance functions. The most widely used covariance functions in machine learning are the squared exponential (2) and the Matérn class (3) covariance functions. This second one is characterized by a smoothness parameter ν; with K_ν the modified Bessel function and $\Gamma(\nu)$ the gamma function.

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \cdot exp\left[\frac{-(\mathbf{x} - \mathbf{x}')^2}{2l^2}\right] \tag{2}$$

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \cdot \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu(\mathbf{x} - \mathbf{x}')^2}}{l}\right)^\nu K_\nu\left(\frac{\sqrt{2\nu(\mathbf{x} - \mathbf{x}')^2}}{l}\right) \tag{3}$$

Hyperparameters are a set of parameters which define completely the covariance function. The squared exponential and the Matérn class covariance function are defined by: 1) σ_f^2 is the maximum allowed covariance; 2) l models the covariance between variables separated a certain distance.

In this work, we define the mean function for the Gaussian Process $m(\mathbf{x}) = 0$, since we do not want to assume a prior data model of the physical process.

3 Sense-Predict-Move Algorithm

We aim to estimate the set of information objects O located in the environment S, with N_A agents. We propose a three stages algorithm: Sense, Predict and Move. Each agent executes the algorithm iteratively at the same time.

3.1 Sense

Let $p^{[i]} = s_{a,b}$ be the spatial location of agent $R^{[i]}$, that measures the information object $o_{a,b} \in O$ with a noise-free sensor. The agent $R^{[i]}$ updates its set of measurements $Z^{[i]}$. The updated set of measurements $\check{Z}^{[i]}$ will be $\check{Z}^{[i]} = Z^{[i]} \cup (o_{a,b}, s_{a,b})$. $Z^{[i]}$ is initialized as an empty set.

After sensing, the agent $R^{[i]}$ shares the set of measurements with the rest of the N_A agents. The updated set of measurements $\check{Z}^{[i]}$ will be $\check{Z}^{[i]} = Z^{[i]} \cup \left\{Z^{[j]}\right\}_{j \in I,\, i \neq j}$.

3.2 Predict

Given the set of measurements $Z^{[i]}$, we predict the information objects in a local area around $p^{[i]} = s_{a,b}$. We use Gaussian Processes for regression.

Scalability and Prediction Area. The computational complexity of the Gaussian Processes for regression grows cubically with the number of observations. Therefore, we define a prediction area centered in $p^{[i]} = s_{a,b}$ to guarantee the scalability when the environment grows. The prediction length is given by R_p. The prediction area $A_{R_p}(a, b)$ is defined as $A_{R_p}(a, b) = \{(r, c)\}$ with $|a - r|, |b - c| \leq R_p$.

For the set of cells $\{(r, c)\} = A_{R_p}(a, b)$, we carry out regression analysis using Gaussian Processes. According to the notation introduced in Section II, we define

[1]For simplicity in the notation, the formulation corresponds to a one-dimensional input space, which can be extrapolated to a multidimensional input space.

the vectors: $\mathbf{f} = [o_{r,c}], \mathbf{x} = [s_{r,c}]$ with $(o_{r,c}, s_{r,c}) \in Z^{[i]}$ and $\mathbf{x}_* = [s_{r,c}]$. Given these vectors, we calculate $\bar{\mathbf{f}}_*$, $\sigma^2_{f_*}$ for the positions $\mathbf{x}_*$ using (1). We assume that the hyperparameters, which define the covariance function, are known. This is a realistic assumption, since we can usually take some measurements and calculate the hyperparameters which maximize the marginal likelihood.

Prediction Information Update. The sets $P^{[i]}$ and $U^{[i]}$ are updated with the measurements $Z^{[i]}$ and predictions $\bar{\mathbf{f}}_*$, $\sigma^2_{f_*}$.

For the set of the cells $\{(r,c)\} = A_{R_p}(a,b)$, we update $P^{[i]}_{r,c} = o_{r,c}$, $U^{[i]}_{r,c} = 0$ if $(o_{r,c}, s_{r,c}) \in Z^{[i]}$. Otherwise we update $P^{[i]}_{r,c}$, $U^{[i]}_{r,c}$ with the corresponding elements[2] of the predicted vectors $\bar{\mathbf{f}}_*$, $\sigma^2_{f_*}$, respectively.

The mean of the Gaussian distribution is the value with the highest likelihood. Therefore, for the locations $s_{r,c}$ on which we do not have measurements, we update $P^{[i]}$ with the mean vector of the predicted distribution $\bar{\mathbf{f}}_*$. We initialize $P^{[i]} = [0]$ and $U^{[i]} = [\infty]$.

3.3 Move

The key concept of the Sense-Predict-Move algorithm lies in the search of high uncertainty locations. We define a reward function which drives an agent following that principle.

We propose two alternative algorithms: Local View and Wide View. Both of them use a greedy approach to maximize the reward function. They differ in how far the agents can see, which leads to the definition of two classes of Motion Graphs. The Motion Graph G_M is an undirected graph which holds the positions to which the agent can move. Each of the nodes in the set of nodes V_{G_M} is associated to a position from the set S. The agent can only move between two nodes if they are linked by an edge, the collection of which is the set of edges E_{G_M}.

The agent calculates its next position optimizing the reward function $H^{[i]}$:

$$H^{[i]}(p^{[i]}, U^{[i]}, S) = \sum_{r=1}^{R_g} \sum_{c=1}^{C_g} f^{[i]}\left(p^{[i]}, U^{[i]}_{r,c}, s_{r,c}\right) \cdot \varphi(s_{r,c}). \tag{4}$$

The function φ allows us to pre-define positions of particular interest, by giving different weights to the spatial positions in S. The agent function $f^{[i]}$ informs us about the reward obtained by the agent $R^{[i]}$ when it moves from $p^{[i]}$ to $s_{r,c}$. We define the agent function as:

$$f^{[i]}\left(p^{[i]}, U^{[i]}, s_{r,c}\right) = \begin{cases} U^{[i]}_{r,c} & s_{r,c} \in N_{G_M}(p^{[i]}), \\ 0 & otherwise, \end{cases} \tag{5}$$

[2] Associated to the same spatial location $s_{r,c}$.

where $N_{G_M}(p^{[i]})$ is the neighbourhood of the node $p^{[i]}$ in the Motion Graph G_M.

The agent's next position $\check{p}^{[i]}$ is calculated according to,

$$\check{p}^{[i]} = \arg\max_{s_{r,c}} H^{[i]}(p^{[i]}, U^{[i]}, s_{r,c}). \tag{6}$$

Local View

The Local View algorithm defines a Motion Graph G_M only based on local knowledge. Let $p^{[i]} = s_{a,b}$ be the agent's location. The set of edges is defined as $E_{G_M} = \{\{s_{r,c}\ ,\ s_{a,b}\}\}$ with $|a-r|, |b-c| \leq 1$ and $(a-r, b-c) \neq (0,0)$. This algorithm lets the agent move to any of the adjacent cells in the environment.

Wide View

The Wide View algorithm tries to imitate the human behavior. We do not restrict the agent's knowledge to a local area, as we do with the Local View algorithm. We let the agent see farther. We guide the agent to high uncertainty positions contained in a wide area restricted by the view length.

For an agent located in $p^{[i]} = s_{a,b}$, the view length R_v defines the view area as $W_{R_v}(a,b) = \{(r,c)\}$ with $|a-r|, |b-c| \leq R_v$.

For the set of the cells $\{(r,c)\} = W_{R_v}(a,b)$, we calculate the highest uncertainty location $s_{ha,hb}$. This location corresponds to the position $s_{r,c}$ in which the variance map element $U^{[i]}_{r,c}$ is the highest.

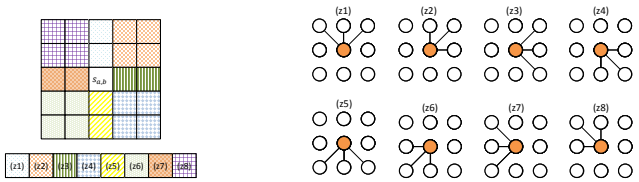
The highest uncertainty position $s_{ha,hb}$ with respect to the agent position $p^{[i]} = s_{a,b}$ will determine the agent's motion graph. Given the agent's location, we define eight zones, in which the highest uncertainty location can be placed. Each of the zones $(z1), (z2), ..., (z8)$ corresponds to a different motion graph. Figure 2(a) represents the different zones for a view area defined by $R_v = 2$. We show in Figure 2(b) a simplified version of the Motion Graph G_M for each of the zones for an agent located in the orange node.

At this point, the agent has already calculated its new location optimizing the proposed reward function. The agent moves to the next location and continues with the next iteration of the algorithm.

Once we stop running the algorithm, each of the robots has measured and estimated different information objects. We are interested in a unique result as output of the algorithm. Our predicted and measured information objects are distributed as $s_{r,c} \sim N(P^{[i]}_{r,c}, U^{[i]}_{r,c})$ with $i \in I$. By using Bayes' rule, we can fuse the Gaussian distributed functions predicted by the N_A agents to obtain the final map.

4 Validation

We test the Sense-Predict-Move algorithm in a hybrid setup including real sensor data and subsequent simula-



(a) (b)

Figure 2: Wide View algorithm. (a): For an agent located in the white cell, we calculate the zone where the highest uncertainty location is placed. Each of the zones corresponds to a different motion graph. (b): Simplified Motion Graph G_M for each of the zones for an agent located in the cell corresponding to the orange node. The agent can move to any of the nodes linked by an edge.

tion of multiple robots' movements and sensing. We use a personal computer running MATLAB[3] for our simulations. The information objects correspond to the magnetic field intensities in an indoor environment. We measure the magnetic field intensities using a robotic platform and a magnetic field sensor.

4.1 Experimental Setup

We use a holonomic robot to generate the magnetic field maps (see Figure 3). The robot is a modified version of the commercially available Slider platform by Commonplace Robotics. Due to its four mecanum wheels the platform is able to perform omnidirectional movements, following input commands for forward, lateral and rotational velocities.

The magnetic field sensor module used in the reported experiments is part of a commercial integrated sensor package (Xsens MTx). We mounted the sensor on a wooden beam that extended 0.75 m from the center of the robot. The purpose of this beam is to separate the sensor from the robot's ferromagnetic components and electromagnetic field generating devices.

We employ a commercial motion capture system (Vicon) to provide ground truth information of the robot's position. Our particular setup consists of 16 infrared sensitive cameras and infrared strobes.

[3]We use the GPML Toolbox [Rasmussen and Nickisch, 2010] to carry out regression with Gaussian Processes.

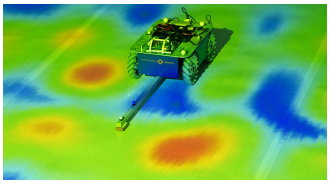


Figure 3: Holonomic robot by Commonplace Robotics, equipped with a magnetic field sensor, generating a magnetic field intensity map. Here, we projected a visual representation of the measured magnetic field intensities on the lab floor.

4.2 Experiments and Discussion of Results

We execute the algorithm with simulated robots. The information objects are based on a $40m^2$ map of the magnetic field intensity of 4128 samples, captured on the ground within the DLR lab with a resolution[4] of 10 cm. The chosen resolution depends on the final application of the obtained map (in this case localization by using the magnetic field features [Angermann *et al.*, 2012]). We learn the hyperparameters of the Gaussian Process model by maximizing the marginal likelihood of a subset of the information objects [Rasmussen and Williams, 2005]. Then we use the remaining information objects for the experimental validation.

We divide the algorithm execution in discrete time steps $k = 1, 2, 3, ...$ Each time step corresponds to one execution of the algorithm loop. To test the algorithm performance, we chose the Exploration Time as the performance metric. We define the Exploration Time as the number of time steps k necessary to reduce the initial mean squared error ($MSE(1)$) by 98%. The MSE in time step k is calculated according to,

$$MSE(k) = \frac{1}{R_g \cdot C_g} \sum_{r=1}^{R_g} \sum_{c=1}^{C_g} \left(o_{r,c} - \bar{P}_{r,c}\right)^2, \qquad (7)$$

where $\bar{P} = \left[\bar{P}_{r,c}\right]_{r=1,...,R_g;\, c=1,...,C_g}$ is the fusion of the N_A combined maps of predictions plus measurements $P^{[i]}$.

For the simulations, we set $\varphi(s_{r,c}) = 1 \ \forall s_{r,c} \in S$. We simulate environments of different sizes by taking subsets of samples of the original magnetic field map. We assume it to be sufficient, as the spatial dependencies of the physical process are uniformly distributed. We analyse the effects on the algorithm performance in the following.

Covariance Function

We carry out experiments with a wide variety of stationary and isotropic covariance functions: squared exponential (2) and Matérn class (3) with $\nu = 3/2$ and $\nu = 5/2$. We test the prediction capabilities of those covariance functions by taking some information objects randomly and predicting the rest of the values in the environment. Figure 4 shows the evolution of the MSE dependent on the percentage of measured information objects. The

[4]Spatial distance between two consecutive measurements.

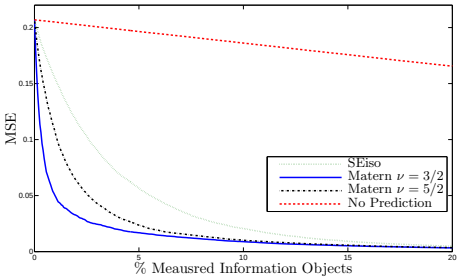


Figure 4: Analysis of the prediction capabilities of the Squared Exponential and Matérn class covariance functions

.

curves correspond to the mean of the *MSE* over 50 trials. It shows, that the Matérn class covariance function with $\nu = 3/2$ offers the best performance. Therefore, the following results correspond to the algorithm using this covariance function.

Prediction and View Length

We aim to characterize the dependency of the prediction and view length in the Exploration Time with environments of different sizes. Our findings are:

- Starting from small values of R_p, the Exploration Time remains quasi-constant as we increase the prediction length R_p.
- Given a physical process, the function $Exploration\ Time = f(R_v)$ is convex. That means, it exists an unique R_v which gives the best algorithm performance.

The parameters R_p, R_v are given by the spatial structure of the physical process and, in consequence, by the hyperparameters which define the covariance function. Therefore, the prediction and view length are independent of the environment's size, which means that the algorithm's complexity is constant in R_p, R_v. We can learn the optimal values of R_p, R_v by initially taking some measurements and running a simulation.

Trajectory

We test the algorithm performance for one agent with a Random trajectory and predefined deterministic trajectories (Meander, Spiral, SplitTwo). The SplitTwo trajectory consists of halving the space consecutively. Because of the spatial structure of the physical process and the Gaussian Processes prediction, we consider that the SplitTwo trajectory could be the fastest alternative to reduce the entropy of the process. We replace the Move part of the algorithm with these trajectories. For the Predict part, we set R_p to cover the complete environment. The simulated environment contains 4128 information objects. We test our algorithm using the Wide

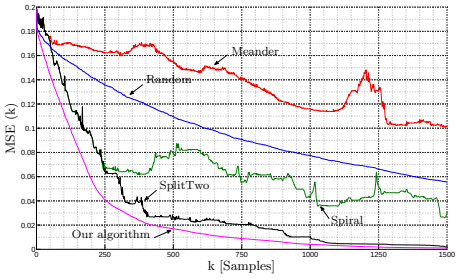


Figure 5: *MSE* evolution for the Agent's Trajectory. We compare the following trajectories: Meander, Random, Spiral, SplitTwo and Our algorithm.

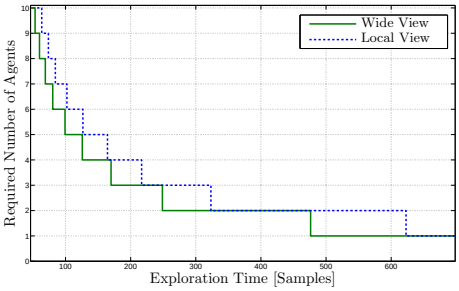


Figure 6: We simulate the algorithm using real data (Magnetic Field Intensity) measured in a DLR lab. Required number of agents to explore a magnetic field intensity map of 1935 samples. We compare the performance of the Local View and Wide View algorithm.

View with $R_v = 5$. Figure 5 shows the mean of the *MSE* evolution over 100 trials. We initialize the agent's starting position randomly for the Random trajectory and Our algorithm trajectory. We remark that only the Random and Our algorithm trajectories can be used in non-rectangular-shaped environments.

Scalability with the Number of Agents

In Figure 6, we compare the number of agents needed to explore the environment with the Local View and the Wide View algorithms. We set R_p to cover the complete environment and $R_v = 5$. We carry out the simulation for a magnetic field intensity map of 1935 samples. It corresponds to the mean Exploration Time over 100 trials with the agents starting in random positions. We observe that we can save one agent just by selecting the Wide View algorithm, without adding an extra complexity. The Exploration Time scales with the required number of agents N_A as $\frac{1}{N_A}$. This property means that we can maintain the Exploration Time in an environment of double size by duplicating the number of agents.

Scalability with the Environment Size

Figure 7 shows the evolution of the Exploration Time as we increase the environment's size. The curve cor-

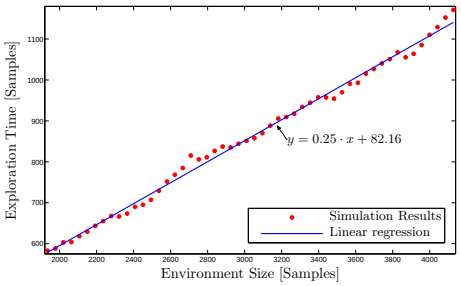


Figure 7: We simulate the algorithm using real data (Magnetic Field Intensity) measured in a DLR lab. Evolution of the Exploration Time with the environment's size.

responds to the mean Exploration Time over 500 trials using one agent starting in random positions, with $R_p = 11$ and $R_v = 5$. The red dots represent the simulation results. In blue we plot the curve we obtain by applying linear regression following a least squares criteria. Considering that the complexity introduced by R_p, R_v is constant and independent of the environment's size (see Section 4.2), the time complexity of the algorithm is $O(M)$ for an environment of M samples.

5 Summary and Conclusions

We have proposed and analyzed a novel algorithm for efficient exploration with a distributed multi-agent system, which does not require any previous information about the environment. We have proved by simulations that the algorithm runs in linear time respect to the number of samples in the environment. Also, it is scalable with the number of agents. We have as well demonstrated the advantage of using Gaussian Processes to model the physical process under study by comparing several covariance functions. Simulations show that the algorithm outperforms the results obtained with several other trajectories. The obtained results in terms of computational complexity suggest that our algorithm could be implemented in a network of low-complexity robots (e.g.swarm of Quadrocopters) to perform the mapping of an environment.

One of our future goals is handling the robots' motion in the presence of obstacles. In this work, we have considered physical phenomena which can be modeled as Gaussian variables. A possible extension of the algorithm includes non-Gaussian physical processes and phenomena which can change over time [Rasmussen and Williams, 2005]. Further steps to improve the algorithm's efficiency include more intelligent path planning algorithms; as well as the online learning and adaptation of the model's hyperparameters according to the field's local characteristics .

References

[Angermann *et al.*, 2012] Michael Angermann, Martin Frassl, Marek Doniec, Brian J. Julian, and Patrick Robertson. Characterization of the indoor magnetic field for applications in localization and mapping. In *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN 2012)*, Sydney, Australia, Nov. 2012.

[Cressie, 1992] Noel Cressie. Statistics for spatial data. *Terra Nova*, 4(5):613–617, 1992.

[Fink and Kumar, 2010] Jonathan Fink and Vijay Kumar. Online methods for radio signal mapping with mobile robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1940–1945. IEEE, 2010.

[Howard *et al.*, 2002] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*, pages 299–308. Springer, 2002.

[Kemppainen *et al.*, 2010] Anssi Kemppainen, Janne Haverinen, Ilari Vallivaara, and J Roning. Near-optimal slam exploration in gaussian processes. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2010 IEEE Conference on*, pages 7–13. IEEE, 2010.

[Krause and Guestrin, 2007] Andreas Krause and Carlos Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *Proceedings of the 24th international conference on Machine learning*, pages 449–456. ACM, 2007.

[Ogren *et al.*, 2004] Petter Ogren, Edward Fiorelli, and Naomi Ehrich Leonard. Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *Automatic Control, IEEE Transactions on*, 49(8):1292–1302, 2004.

[Olfati-Saber *et al.*, 2007] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[Poduri and Sukhatme, 2004] Sameera Poduri and Gaurav S Sukhatme. Constrained coverage for mobile sensor networks. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 165–171. IEEE, 2004.

[Rachlin *et al.*, 2005] Yaron Rachlin, John M Dolan, and Pradeep Khosla. Efficient mapping through exploitation of spatial dependencies. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005*

IEEE/RSJ International Conference on, pages 3117–3122. IEEE, 2005.

[Rasmussen and Nickisch, 2010] Carl Edward Rasmussen and Hannes Nickisch. Gaussian processes for machine learning (gpml) toolbox. *The Journal of Machine Learning Research*, 11:3011–3015, 2010.

[Rasmussen and Williams, 2005] Carl Edward Rasmussen and Christopher KI Williams. Gaussian processes for machine learning (adaptive computation and machine learning). 2005.

[Singh *et al.*, 2009] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J Kaiser. Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, 34(2):707, 2009.

[Stranders *et al.*, 2008] Ruben Stranders, Alex Rogers, and Nick Jennings. A decentralized, on-line coordination mechanism for monitoring spatial phenomena with mobile sensors. 2008.

[Williams and Sukhatme, 2012] Ryan K Williams and G Sukhatme. Probabilistic spatial mapping and curve tracking in distributed multi-agent systems. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1125–1130. IEEE, 2012.